Opmærkning og opmærkningssprog - om HTML og XML
Introduktion til og diskussion af opmærkning med HTML og XML

Lidt om hjemmesider og opmærkning

Vi har alle adskillige gange besøgt en hjemmeside i vores jagt på informationer når vi søger oplysninger om personer, produkter eller ord og begreber som vi ikke forstår. Vi er så vant til at bevæge os i cyberspace og til at besøge hjemmesider, at vi ikke opfatter det som anvendt teknologi, ikke tænker over hvad der gemmer sig bag en hjemmeside og hvordan den er bygget op. Men vi har nogle ganske præcise forventninger til en hjemmeside: den skal indeholde de informationer vi søger, og de skal være let tilgængelige. Derfor er vi heller ikke sene til at fælde dom over en hjemmeside hvis vi bliver skuffede: ”Den er kedelig, den gider jeg ikke besøger igen!”, ”Den indeholder ikke de oplysninger jeg søger!”, ”Den er svær at bruge!”.

En hjemmeside er et udstillingsvindue, det sted hvor en leverandør præsenterer sine produkter for sine kunder, og stedet hvor man gerne skulle få nye kunder. Det er derfor vigtigt at en hjemmeside på en og samme tid er smart, indbydende, let at bruge og informativ. Det stiller store krav til **design**, til opbygning og indretning af hjemmesiden. De centrale informationer skal placeres det rigtige sted på siden – eller siderne – og det skal være nemt at bevæge sig rundt og komme fra et sted til et andet på en hjemmeside. Og det hele skal samtidig være overskueligt, ligesom der skal være en naturlig sammenhæng mellem informationer som præsenteres samme sted.

Når man leder efter en hjemmeside, bruger man en **browser***[footnote]*, et program som kan finde og fortolke indholdet på en hjemmeside. Det sprog man anvender når man bygger sin hjemmeside, skal derfor helst være et sprog der ikke er afhængigt af at bruger anvender en bestemt type computer eller en bestemt browser. Lige netop sådan et sprog er HTML, en forkortelse der dækker over **HyperText Markup Language.**
En af de mest anvendte browsere er *Interne Explorer* fra Microsoft

At en hjemmeside er *HyperText* vil sige, at det er tekst som ikke er bundet til papir, ikke er bundet af en lineær præsentation og læsning fra venstre mod højre. Man kan derimod bevæge sig frit i teksten, udnytte links og henvisninger og springe fra et sted i teksten til etandet ved at klikke på et ord, en illustration eller på en adresse til en anden hjemmeside.

Om opmærkning og opmærkningssprog

At HTML er et **Markup Language** vil sige at det er et sprog som man bruger til at forsyne en tekst med markører. Vi siger også at HTML er et *opmærkningssprog*.

Helt konkret og bogstaveligt går opmærkning derfor ud på at indsætte markører (eng. *tags*) i en tekst. Bruger vi HTML som opmærkningssprog, kan vi slå mængden af mulige markører og deres betydning op i en særlig HTML-ordbog.

Opmærkning svarer på mange måder til vores brug af et tekstbehandlingssystem når vi ønsker at fremhæve og udskille vigtige dele i en tekst som for eksempel i:

Figur `Eric T. Ray Learning XML O’Reilly 2003`

Vi bruger som noget helt naturligt forskellige indbyggede funktioner i det aktuelle tekstbehandlingssystem i vores opmærkning, det kan være fed skrift eller *italic* og vi kan ændre på størrelsen af bogstaverne. Men mange gange bruger vi også eksplicitte markører, som for eksempel i:

Figur `Forfatter: Eric T. Ray Titel: Learning XML Forlag: O’Reilly 2003 Udg.: 2003`

I ovenstående eksempel indsætter vi markører, **Forfatter, Titel, Forlag** og **Udg.**, for at gøre det lettere for læseren at forstå hvad det er for en type informationer der vises.

HTML og XML

I det foregående afsnit er begrebet *opmærkning* introduceret og HTML er beskrevet som et *opmærkningssprog*. To eksempler, fig.1 og fig.2 illustrerer hvordan man kan bruge funktionerne i et tekstbehandlingssystem, fed skrift, skråskrift og skrifttyper, til opmærkning. De viser tillige at funktioner som 'indrykning' og 'tekstbox' også fungerer som en form for opmærkning, dvs som et middel til rent grafisk at fremhæve og tydeliggøre det centrale i den information som man formidler. De to eksempler illustrerer en yderligere og væsentlig forskel på måden at opmærke på. I fig. 2 er der indsat markører som beskriver indholdet i den information som præsenteres. Vi kan direkte aflæse af markørerne at 'Eric T. Ray' er navnet på en forfatter og at 'Learning XML' er titlen på en bog som han har skrevet. Denne information mangler i fig. 1, som alene fokuserer på den grafiske præsentation og på at adskille forfatter, titel og forlag fra hinanden ved hjælp af forskelle i skriften.

Denne forskel i fokus – præsentation over for indhold – genfinder vi i forskellen mellem HTML og XML, **eXtensible Markup Language**.

Hvad vil du fremhæve: præsentation eller indhold

HTML har som opmærkningssprog fokus på *præsentationen af informationer*. Det er derfor det sprog vi bruger når vi bygger hjemmesider. XML har fokus på *indholdet*. Det svarer ganske godt til forskellen mellem fig. 1 og fig. 2. Vi bruger XML og de markører vi sætter ind i teksten til at sige noget om indholdet: hvad handler det om, som i fig. 2, og modsat bruger vi HTML når vi lægger vægt på hvordan teksten skal fremstå og præsenteres.

Vi kan illustrere forskellen på følgende vis:

Figur

```
<b> Eric T. Ray </b>
<i> Learning XML </i>
<p> o’Reilly </p>
<font size=’-1´> O’Reilly 2003 </font>
```

I fig. 3 – lad os kalde det et HTML-dokument – er der brugt markører fra HTML til at præsentere en streng med fed skrift (l.1), med skråskrift (l.2) og til at nedsætte størrelsen på den anvendte skrifttype (l.4). Læg mærke til at en markør (eng. *tag*) er omgivet af særlige tegn, tegnet < som starten på en markør, og > som afslutning. Markøren angiver at skriften skal være fed (eng.*bold*), og markøren <i> angiver at skiften skal fremstå som *italic*. Med markøren <p> angives at der er tale om ren tekst, en (text)paragraph. Læg endvidere mærke til at en opmærkning altid bør omfatte en **startmarkør** og en **slutmarkør**. Slutmarkøren kender vi på syntaksen som for eksempel i . Opmærkningen ... har således den effekt at alt hvad der står mellem startmarkøren og slutmarkøren bliver præsenteret med fed skrift.

Sammenfattende kan man sige, at når man opmærker i HTML, fungerer man som både en form for grafiker og som typograf. Når vi opmærker i XML, er fokus et andet. Da er det indholdet det gælder:

Figur

```
<forfatter> Eric T. Ray </forfatter>
<titel> Learning XML </titel>
<forlag> O’Reilly </forlag>
<udg> 2003 </udg>
```

De markører som er brugt i fig. 4 – lad os kalde det et XML-dokument – siger alene noget om hvad det er for en type information der står mellem de to markører, hvad data mellem start- og slutmarkør handler om.

Vi kan sammenfattende karakterisere forskellen mellem HTML og XML som en forskel mellem valg af *design* , HTML, og valg af *information*, XML. Vi har på den ene side de data som vi vil præsentere, på den anden vores viden om de informationer som er repræsenteret i data og vores forestilling om hvordan vi kan *modellere* disse informationer i XML. En bog er skrevet af en person, det vi normalt kalder FORFATTER, den har en TITEL og er udgivet af et FORLAG på et bestemt tidspunkt. Det er denne vores idé og forestilling om den typiske bog og tilhørende informationskategorier som vi fokuserer på i XML og sætter navn på ved hjælp af vores markører.

Det siger sig selv, at en browser ikke ved hvilke informationskategorier vi har i hovedet og anvender når vi opmærker i XML, uanset hvad vi opmærker. Modsat er det praktisk at den kan genkende og fortolke de markører som definerer præsentationen. Derfor er mængden af markører i HTML og deres betydning fastlagt på forhånd. Vi kan ikke lave om på dette og introducere vores egne markører[footnote].
Fremover vil vi anvende den gængse engelske terminologi for markører: tags

XML derimod giver brugeren mulighed for lave og bruge sit eget opmærkningssprog i og med at vi skiller indholdet fra præsentationen. Et XML-dokument er **strukturerede data**, men også **struktureret information**. Strukturen er identisk med de tags der anvendes. De afspejler den informationsstruktur somer indeholdt i de data der opmærkes som for eksempel i tilfældet med en bog. Hvis vi udveksler vores tags eller informationskategorier med hinanden, vil vi også kunne hente, bruge og genbruge oplysninger med samme struktur, og vi kan uden problemer vise det hele hvis det er pakket ind i HTML.

Test dig selv

Som afslutning på dette første modul følger et par opgaver som du kan bruge til at træne opmærkning.

Som det første skal du samle alle de oplysningstyper – informationskategorier – som efter din mening typisk er knyttet til beskrivelsen af en bog.

Lav en liste med de relevante begreber (forfatter, titel, forlag.....) og lav derefter en liste hvor begreberne er noteret som tags.

Nedenfor finder du tre forskellige beskrivelser af tre forskellige bogtitler.

Sammenlign dine liste med de tre beskrivelser og udvid om nødvendigt listerne med nye kategorier, kategorier som du ikke har tænkt over i første omgang.

Kopiér en af teksterne over i dit tekstbehandlingssystem, indsæt dine tags i teksten og gem den som et XML-dokument, det vil sige som en fil med efternavnet xml, for eksempel: jørgen_leth.xml

Example:
Tekst 1: hentet hos Gentofte Kommunes Biblioteker:
Roman Anne Marie Løn. Sekstetten : roman / Anne Marie Løn. - [Kbh.] : Gyldendal, 2008. - 550 sider ISBN 978-87-02-07227-3 : hf. : kr. 299,00. ISBN 978-87-02-07365-2 : ib. : kr. 349,00.

Example:
Tekst 2: hentet hos Gentofte Kommunes Biblioteker:
Lydbog, Biografi. Det uperfekte menneske / Jørgen Leth
Gyldendal Lyd, 2008-Forfatteren, filminstruktøren og Tour de France-eksperten Jørgen Leths (f. 1937) erindringsessays, hvori han beskriver øjeblikke i sit liv og omkostningerne ved at leve som æstetiker: rastløsheden, depressionerne og ensomheden.

Example:
Tekst 3: hentet hos Gyldendals Bogklub (og derfor er det ikke sikkert at alle informationerne er relevante i din model!):
Jo Nesbø
Flagermusmanden
Klubpris: 149,-
Første bind i Jo Nesbøs suveræne krimi-serie om Harry Hole
Den norske kriminalbetjent Harry Hole sendes til Australien for at bistå i opklaringen af mordet på en ung, norsk kvinde. I lufthavnen mødes han af Andrew kensington, en aboriginal kriminalbetjent med en mindst ligeså broget fortid som ham selv. Sammen bevæger de sig rundt i Sydneys red light district, King's Cross, og forsøger at navigere efter de svage ekkoer fra en morder. Oversat af Allan Hilton Andersen efter Flaggermusmannen

Afslut denne første øvelse med at udskrive din færdige og endelige liste over informationskategorier og tilhørende tags knyttet til beskrivelsen af en bog.

Hvordan vil du forklare forskellen i følgende udsagn:

HTML er strukturerede data

XML er struktureret information

XML - grundlaget
En gennemgang af det elementære grundlag for opmærkning i XML og opbygningen af et XML-dokument

XML – GRUNDLAGET

1. XML – før vi starter

1.1 XML – eXtensible Markup Language

XML præsenteres generelt som et opmærkningssprog på lige fod med HTML. Der er altså i begge tilfælde tale om en form for sprog, de hedder begge **Markup Language**, og de bruger begge **tags** til opmærkning. Den samlede mængde af tags kaldes traditionelt for et **vokabular**.

I XML udarbejder du dit eget vokabular. I HTML er der et foruddefineret vokabular, det vil sige en mængde gyldige, navngivede tags, som angiver hvordan data i et HTML-dokument skal præsenteres. Derfor vil man også møde de samme tags i stort set alle HTML-dokumenter. Formålet med et prædefineret vokabular er indlysende: en hvilken som helst browser kan genkende og fortolke de tags der forekommer i et dokument og vise indholdet i dokumentet i overensstemmelse hermed.

Et XML-vokabular udgøres af tags som er valgt og defineret med henblik på at fortælle noget om indholdet i data. Et XML-vokabular vil derfor være forskelligt fra det ene XML-dokument til det andet, afhængigt af de skiftende indhold som skal opmærkes. Der findes imidlertid regler for hvordan et XML-vokabular skal se ud, hvordan det skal struktureres. Disse regler – se dem i afsnit 3 i dette modul – udgør rygraden i XML.

XML er et sæt *regler* for opmærkning. Følger man disse regler bliver

resultatet et velformet XML-dokument som gør det muligt at bygge,

vedligeholde, udveksle og genbruge data. [footnote]

“Strictly speaking, XML is not a markup language. A language has a fixed vocabulary and grammar, but XML doesn’t actually define any elements. Instead, it lays down a foundation of syntactic constraints on which you can build your own language.” Ray: *Learning XML*, O’Reilly 2003, p.6

XML er således et **meta-sprog**, det vil sige et sprog som kan bruges af alle der gerne vil opmærke en eller flere tekster i deres eget opmærkningssprog. Men hvis man vil gøre sig forståelig for omverden, for eksempel fordi man vil distribuere sine opmærkede dokumenter, er det vigtigt at man overholder den særlige syntaks der gælder for XML.

1.2 XML-dokumentet

En opmærket tekst gemmes som et *XML-dokument*. Et XML-dokument indeholder strukturerede data som i kraft af opmærkningen udgør *struktureret information*.

Vi vil typisk opbevare et XML-dokument som en fil, en fysisk enhed gemt i et lager på en computer, hvorfor vi giver vores fil efternavnet *.xml. Men vi vil også gerne have at dokumentet, den logiske enhed, *informationsenheden*, kan læses og genkendes som XML af en XML-processor. Det vil blandt meget andet sætte os i stand til at udveksle XML-dokumenter med andre brugere på tværs af computere, styresystemer og diverse andre teksnologiske restriktioner. For at opnå dette bliver nødt til at følge de regler som definerer hvad der gør strukturerede data til et **velformet** og **gyldigt** XML-dokument.

At et dokument er *velformet* vil sige, at det overholder de syntaksregler der er defineret for XML. At et dokument er *gyldigt* vil sige, at der foreligger en form for grammatik for opmærkningen og at den konkrete opmærkning er i overensstemmelse hermed.

En sådan grammatik, som definerer hvilke tags der må forekomme, hvor mange gange og i hvilken rækkefølge, kaldes her og i tilhørende moduler også for en informationsmodel [footnote]. Den særlige *syntaks* som gælder for opbygning af informationsstrukturerne i et XML-dokument vil blive

gennemgået i det følgende. Ønsker man at studere dette emne i en original version, anbefales det at starte på www.w3.org.
Begrebet *informationsmodel* bruges her om den konceptuelle pendant til det som med en teknisk term i XML betegnes som en DTD: Document Type Definition. I modul 3 gennemgåes dette nærmere.

2. Opmærkning med XML

2.1 Forlægget

Det er som det første vigtig at være opmærksom på, at man *ikke* skal skrive sine XML-dokumenter i et tekstbehandlingsprogram. Et sådant program indsætter koder overalt i dokumentet, for linieskift, indrykning af afsnit og andet tilsvarende, og den XML-processor som skal tage hånd om dokumentet vil ikke acceptere disse koder. Det vil give en masse fejlmeddelelser. Brug derfor enten en almindelig teksteditor som for eksempel Notepad, eller find en XML-editor på nettet [footnote].
Opmærkninger vist i dette og de øvrige moduler er foretaget ved hjælp af den særlige XML-editor Altova XMLSpy.

Udgangspunkt for gennemgangen af de basale byggeklodser i XML er følgende bogtitel fundet i Gentofte Kommunes online-katalog og her præsenteret i let bearbejdet version:

Figur `Roman Anne Marie Løn Sekstetten [Kbh.] : Gyldendal, 2008. - 550 sider ISBN 978-87-02-07227-3 : hf. : kr. 299,00. ISBN 978-87-02-07365-2 : ib. : kr. 349,00.`

Skal denne information omsættes til et XML-dokument, kunne resultatet være som

vist nedenfor:

1: <?xml version="1.0" encoding="UTF-8" ?>

2: <bog>

3: <forfatter>

4: <fornavn>Anne Marie</fornavn>

5: <efternavn>Løn</efternavn>

6: </forfatter>

7: <titel>Sekstetten</titel>

8: <forlag sted="København">Gyldendal</forlag>

9: <udg>2008</udg>

10 <sidetal>550</sidetal>

11: <ISBN>

12: <heftet pris=" 299,00">978-87-02-07227-3</ heftet>

13: <indbundet pris="349,00">978-87-02-07365-2</ indbundet >

14: </ISBN>

15: <genre>roman</genre>

16:</bog>

fig. 2: sekstetten.xml

2.2 XML – sproget

Et XML-dokument består af en **prolog** og et antal **elementer**. I en prolog placerer man alle de instrukser som specificerer hvordan dokumentet skal

behandles, for eksempel en grammatik for dokumentet. Her angiver man også om for eksempel et style-sheet skal inddrages i den aktuelle kørsel. Prologen er altid det første man møder i et XML-dokument.

2.2.1 Prologen

Som det allerførste i linie 1 i sekstetten.xml er der i prologen placeret en såkaldt **deklaration**:

1: <?xml version="1.0" encoding="UTF-8" ?>

Her defineres at der er tale om XML og at der anvendes version nummer 1.0. Endvidere defineres et tegnsæt og dermed mængden af tegn som processoren kan genkende[footnote]. I denne mængde indgår blandt andet de danske tegn *æ, ø, å* og det gør dermed tilværelsen lettere for alle dansktalende brugere af XML.
UTF-8 = USC Transformation Format (Unicode Standard Characters) for 8 bit

Anvendelsen og placeringen af spørgsmålstegn definerer en såkaldt *processing instruction* (PI), det vil sige at alle informationer her er systemorienterede, at de ikke indgår i selve XML-dokumentet og derfor ikke skal godkendes som velformede af XML-processoren.

Resten af dokumentet er bygget op ved brug af de grundlæggende og klassiske byggeklodser i XML, som introduceres nedenfor.

2.2.2 Elementer

Den grundlæggende byggeklods i XML er *elementet*. Et element består af en **starttag**, for eksempel <fornavn>, og en tilhørende **sluttag**, </fornavn> som vist i linie 4. En sluttag kender vi på tegnet '/'. Mellem disse tags finder vi elementets **indhold**, forfatterens fornavn. Et element i XML udgøres altså af en starttag, af et indhold, og af en sluttag. Ud fra denne betragtning kaldes et element ofte for en *container*.

Et element skal have et **navn**. Det er dig som opmærker der bestemmer, hvad elementet skal hedde, og det er ikke mindst på dette punkt at XML

adskiller sig fra HTML. Antallet af navngivne elementer udgør det samlede XML-vokabular.

I sekstetten.xml finder vi i linie 4 og 5 elementerne FORNAVN og EFTERNAVN [footnote]:
Når vi fremover i teksten omtaler elementer, skrives elementets navn i versaler

4: <fornavn>Anne Marie</fornavn>

5: <efternavn>Løn</efternavn>

Elementer af denne art kalder vi for **tekstelementer**, da indholdet i elementet er ren tekst. Indholdet i FORNAVN er tekststrengen ’Anne Marie’, og tilsvarende er indholdet i elementet EFTERNAVN tekststrengen ’Løn’.

I linie 3 finder vi en starttag til elementet FORFATTER, den tilhørende sluttag står i linie 6. Lad os se lidt nærmere på indholdet i dette element:

3: <forfatter>

4: <fornavn>Anne Marie</fornavn>

5: <efternavn>Løn</efternavn>

6: </forfatter>

Elementet FORFATTER indeholder ikke tekst, men derimod tekstelementer, elementerne FORNAVN og EFTERNAVN. Vi siger at disse elementer er *indlejret* (eng. embedded) i elementet FORFATTER.

Et element der som indhold har et eller flere andre elementer, kalder vi for et **strukturelement**.

Strukturelementer skal, som navnet angiver, skabe struktur i informationsmængden. Strukturen kan vi i dette tilfælde definere som *en relation*, en relation mellem elementer og anførte informationer: at en

forfatter er en person som har et fornavn og et efternavn. Eller, på en anden måde, at FORNAVN og EFTERNAVN er egenskaber ved en FORFATTER.

Der findes et yderligere og specielt strukturelement i dokumentet: elementet BOG. Ser vi nærmere efter, viser det sig at *hele dokumentet*, fra linie 2 til og med linie 16, er ét enkelt element, eller én enkelt container kaldet BOG. Dette element indeholder hele det samlede XML-dokument. Et element som indeholder det samlede XML-dokument, kalder vi for dokumentets **rodelement**.[footnote] Vi skal senere se hvordan man kan udnyttet rodelementet til at få fat i hele indholdet i et XML-dokument.
Når vi betegner dokumentet som den samlede mængde elementer fra linie 2 til og med linie 16, skyldes det at XML-deklarationen er optionel, at den altså strengt taget ikke er nødvendig.

Der findes en tredie form for element: det tomme element. Der er ikke noget tomt element i vores dokument. Vi skal senere se nogle tilfælde, hvor det er nyttigt at gøre brug af tomme elementer.

2.2.3 Attributter

Elementet FORLAG er udformet på en speciel måde. Der er tale om et tekstelement, en container, som vi bruger til at anføre navnet på det forlag som er ansvarlig for udgivelsen. Men i elementets starttag er der indbygget mere information, en angivelse af hvor det aktuelle forlag findes:

8: <forlag sted="København">Gyldendal </forlag>

Information angivet på denne måde i en starttag kalder vi for et **attribut**.

Et attribut er generelt en karakteristisk bestemmelse af en person eller et begreb. Et attribut i XML er supplerende information som karakteriserer det element som attributtet er knyttet til og eventuelle data indeholdt i elementet. Det er med andre ord data om data, det vi også kalder for **metadata**. I det aktuelle tilfælde, i linie 8, er det data som karakteriserer forlaget med hensyn til beliggenhed, at der er tale om forlaget Gyldendal i København, og ikke Gyldendal i Oslo.

Et attribut skal have et **navn** – navnet bestemmer du – og en tilhørende **værdi**. Det er værdien som specificerer data. Et forlag, alle forlag, er hjemmehørende et bestemt sted, og det konkrete forlag Gyldendal ligger i København:

sted=”København”

Navnet på dette attribut er ’*sted* ’og værdien knyttet hertil er ’*København*’. Læg mærke til at man anvender tegnet ’=’ til at forbinde navnet på et attribut med attributets værdi, og at værdien er anført i dobbelte citationstegn.

I sekstetten.xml er der ligeledes brugt attributter i elementerne HEFTET og INDBUNDET til at angive to forskellige priser (linie 12 g 13). Hvornår man skal placere sine informationer i et element og hvornår de hører hjemme som et eller flere attributter, er en afgørelse som træffes af den der opmærker og som ofte er genstand for megen diskussion. Man kan for eksempel diskutere, om SIDETAL (linie 10) skal være et selvstændigt element – kan man på et tidspunkt have glæde af denne oplysning som selvstændig enhed ? – eller om det burde optræde som attribut. Og hvor skulle det så placeres ?

De øvrige elementer i dokumentet giver ikke anledning til yderligere kommentarer. Alt i alt er der således

3 strukturelementer: BOG, FORFATTER og ISBN

9 tekstelementer: FORNAVN, EFTERNAVN, TITEL, FORLAG, UDG (=udgivelse), HEFTET, INDBUNDET, SIDETAL, GENRE

3 attributter: STED, HEFTET, INDBUNDET

3. Syntaks

For at kunne tale om et velformet XML-dokument, skal der gælde følgende:

1. Et XML-dokument bør starte med en XML-deklaration, og denne skal i så fald være placeret som det allerførste i dokumentet
2. Et XML-dokument skal indeholde ét element som indeholder samtlige andre elementer i dokumentet. Dette element kaldes for rodelementet
3. Et element i XML skal have både en starttag og en sluttag
4. Navnet på start- og sluttag skal matche fuldstændigt
5. Navnet i en tag må ikke indeholde blanke tegn (mellemrum), men man kan i stedet bruge underscore
6. Navnet på en tag må ikke begynde med et tal
7. Værdierne anført i et attribut skal anføres i dobbelte citationstegn (jf ovenfor)

At et XML-dokument er *velformet* vil sige, at ovennævnte regler er overholdt. Der findes imidlertid en række typiske fejl, fejl som man måske ikke selv får øje på i første omgang, men som fremkommer som en fejlmeddelelse fra processoren når man vil have testet om et dokument er velformet. Regel nummer 4 er nok den regel man lettest kan komme til at forsynde sig imod. Men heldigvis er xml-processoren ekstremt sensitiv, og den vil give besked på skærmen når der er tale om en syntaksfejl og angive nummeret på linien i dokumentet hvor der er en fejl.

De følgende eksempler overholder ikke reglerne 4-6 (og er derfor forsynet med en *):

1: * <Fornavn> Niels </fornavn>

2: * <fornavn> Niels </navn>

3: * <titel> Sekstetten <titel>

4: * <ISBN nummer> 123456789 </ISBN_nummer>

5: * <6_række> 6 </6_række>

Der er tale om følgende fejl:

1: der er brugt ’F’ og ’f’ i navnet, de matcher ikke

2: <fornavn> matcher ikke med </navn>

3: tegnet '/' mangler i sluttag

4: der er en blank i starttag

5: navnet på en tag må ikke begynde med et tal

Udover syntaksfejl forekommer en anden type fejl, fejl som ikke fanges af processoren, fordi der er tale om *logiske fejl*. Logiske fejl hænger sammen med opmærkningen og opbygningen af dokumentet. Det kan for eksempel være, at man fejlagtigt anfører forfatterens fornavn som efternavn og efternavnet som fornavn. Den slags fejl skal fanges af den menneskelige processor.

4. Test dig selv

- I sekstetten.xml er GENRE et selvstændigt element med indholdet *roman*. Denne information kan også anføres som et attribut. Hvor vil du placere dette attribut ?
- Giv et forslag til opmærkning af følgende bogtitel:

Torgny Lindgren

Norrlands Akvavit

Samleren 2008

org.: Norrlands Akvavit 2007

overs.: Karsten Sand

Torgny Lindgren (f. 1938) er en af Sveriges helt store forfattere og desuden medlem af Det Svenske Akademi, som uddeler Nobelprisen i litteratur. Hans forfatterskab tæller titler som Batseba, I brogede Blades vand, Pölsa og Dorés Bibel.

416 sjæle! 416 sjæle frelste Olof Helmersson for 50 år siden i et dalstrøg i Västerbotten. Med livfulde ord og store armbevægelser, inderlighed og harmonikaspil ledte han sin flok. Helmersson var vækkelsens konge. Tanken om de 416 sjæle har boet i ham lige siden, også da han forlod de helliges forsamling og fik job på et sindsygehospital. Og tanken har plaget ham. Så meget, efterhånden, at der skal gøres op. Så den 83-årige Helmersson vender tilbage og slår sig ned hos Ivar og Asta og gør sig klar til at møde de vakte. Men det går ikke helt efter planden og Helmerssons hoved.

Fra Informationsmodel til en DTD
En præsentation af opbygning og brug af en DTD med udgangspunkt i en informationsmodel

DTD – DOCUMENT TYPE DEFINITION

1. Dokument – informationsmodel – DTD

Udgangspunkt for fremstillingen i dette modul er det **XML-dokument** som er vist i fig.1 nedenfor. Vi vil gerne sikre os, at dokumenten er både *velformet* og *gyldigt*. **Velformet** er dokumentet, når det overholder de syntaksregler som gælder for XML *[footnote]*. At et XML-dokument er **gyldigt**, vil sige at det er i overensstemmelse med en foreliggende standard for opmærkning, den standard som kaldes en **DTD, en Document Type Definition**.
jf. http://www.w3.org/TR/xml/

En DTD fastlægger

- hvilke elementer der forekommer i et XML-dokument
- hvor mange gange det enkelte element forekommer
- i hvilken rækkefølge elementerne forekommer

I det følgende vil vi (1) anskue en DTD som en implementeret **informationsmodel**, dernæst (2) se nærmere på reglerne for hvordan man skal bygge og notere informationerne i en DTD, og endelig (3) se nærmere på hvordan en DTD bruges sammen med et XML-dokument.

2. Fra dokument til informationsmodel

2.1 Opmærkningen

Udgangspunkt er opmærkningen af nedenstående lydbog (ex1):

Figur

```
<online_katalog>

  <bog>

    <forfatter nation="dk">

      <fornavn>Jørgen</fornavn>

      <efternavn>Leth</efternavn>

    </forfatter>

    <impressum>

      <titel>Det uperfekte menneske</titel>

      <forlag sted="København">Gyldendal</forlag>

      <udgivet>2007</udgivet>

      <ISBN format="lydbog" volumen="11T:36M">978-
87-02-06068-3</ISBN>

      <pris>kr. 298</pris>

    </impressum>

    <stof>

      <handling>Cool, elegante, morsomme og
selvudslettende
                      historier fra et uperfekt
liv med litteratur,
```

```
                      film, sport, kvinder og
rejser. Digteren,
                      filminstruktøren og Tour de
France-kommentatoren
                      m.m. har skrevet en
erindringsbog af de sjældne,
                      som samtidig tegner et
portræt af dansk
                      kulturliv gennem de sidste
40 år. Jørgen Leths
                      karakteristiske stemme fører
lytteren gennem en
                      af dette årtis mest omtalte
bøger.</handling>

      <genre>biografi</genre>

      <original_titel> Det uperfekte Menneske
</original_titel>

    </stof>

  </bog>

</online_katalog>
```

Denne opmærkning vil vi gerne bruge som standard ved opmærkning af bogtitler. Det er derfor vigtigt at give en beskrivelse af denne standard således at alle nye og tilkomne titler følger samme opmærlning. Det kræver at der udarbejdes en oversigt over hvilke elementer og attributter der indgår i opmærkningen. En sådan oversigt kaldes også for en **informationsmodel** og har vi først uadarbejdet en informationsmodel, kan vi bruge denne som grundlag for at udvikle en DTD.

2.2 Informationsmodellen

Informationsmodellen kommer til at se således ud (ex2):

```
1. BOG prototypen – strukturelement, container for
den samlede beskrivelse af en bog

  1.1 FORFATTER strukturelement; indgang til
informationer om den
      aktuelle forfatter

      ATTRIBUT: nation: angiver forfatters
nationalitet

      1.1.1 FORNAVN tekstelement; indeholder
forfatterens fornavn

      1.1.2 EFTERNAVN tekstelement; indeholder
forfatterens efternavn

  1.2 IMPRESSUM strukturelement; indgang til alle
informationer
      om bogens udgivelse

      1.2.1 TITEL tekstelement; indeholder bogens
titel

      1.2.2 FORLAG tekstelement; indeholder navnet
på forlaget

            ATTRIBUT: sted: angiver hjemsted for
forlaget
```

1.2.3 UDGIVET tekstelement; indeholder året for udgivelsen

1.2.4 ISBN tekstelement; indeholder bogens ISBN-nummer

ATTRIBUT: format: angiver udgivelsens format:
”heftet”, ”indbundet” eller ”lydbog”

ATTRIBUT: volumen: angiver sidetal eller varighed
for en lydbog

1.2.5 PRIS tekstelement; indeholder indkøbspris

1.3 STOF strukturelement; indgang til informationer om en
bogs handling, genre og orignaltitel

1.3.1 HANDLING tekstelement; indholder et indholdsresume,
typisk svarende til indholdet angivet på bogens bagside

1.3.2 GENRE tekstelement; indeholder angivelse af genre

1.3.3 ORIGINAL_TITEL tekstelement; indeholder angivelse
af originaltitel, er specielt relevant for oversat litteratur

Med denne oversigt, det vi kalder en formaliseret informationsmodel, har vi dannet os et samlet overblik over elementerne i modellen, over relationerne mellem elementerne, og over mulige attributter i et standarddokument. Enhver ny bog som skal indgå i ONLINE_KATALOG skal beskrives og opmærkes som XML-dokument i henhold til modellen i ex2.

Næste skridt bliver at implementere beskrivelsen i form af en DTD som vi kan bruge til at **validere** alle nye beskrivelser. At validere et dokument vil sige at fastlægge om der er tale om et gyldigt XML-dokument, et dokument som lever op til standarden.

3. DTD: en implementeret standard for opmærkning

En DTD, en *Document Type Definition*, er en standard, en **deklaration**, som angiver *hvilke navngivne elementer* der kan forekomme i en bestemt type XML-dokumenter, *hvor mange gange* elementerne forekommer, og *i hvilken rækkefølge*. På denne måde deklarerer og standardiserer en DTD et særligt **vokabular**, mængden af navngivne elementer og eventueller attributter, og en særlig struktur, relationen mellem elementer, for forekomster af en type XML-dokumenter.

3.1 Syntaks

Der gælder en særlig syntaks for opbygningen af en DTD (jf. http://www.w3.org/TR/2008/REC-xml-20081126):

1. Alle elementer i en DTD beskrives inden for tegnene < og >, samme notation som gælder for angivelsen af tags i XML.
2. Med tegnet ’!ELEMENT’ angives at der er tale om en PI.
3. Betegnelsen ELEMENT er et reserveret ord i XML, forbeholdt denne funktion i en DTD.

En DTD indeholder en beskrivelse, en **elementdeklaration**, af alle elementer i et tilhørende XML-dokument efter følgende skabelon:

<!ELEMENT elementnavn (indhold) >

Som det første i en elementdeklaration anføres navnet på det element som skal deklareres. Læg mærke til at navn og ortografi for hvert enkelt element i en DTD skal matche den tilsvarende angivelser i XML-dokumentet og vice versa. Efter elementets navn anføres i parentes elementets indhold. Ved at se på indholdet i parentesen kan man konstatere om der er tale om et strukturelement eller et tekstelement.

Det vil blive illustreret med følgende eksempler:

3.1.1 Rodelementet

<!ELEMENT online_katalog ((bog)+) >

Der findes ikke specielle regler for rækkefølgen af elementdeklarationer i en DTD. Men det kan anbefales at følge rækkefølgen i den formaliserede informationsmodel. Denne rækkefølge svarer til rækkefølgen i det tilhørende XML-dokument, og ved at følge denne er risikoen for at glemme eller overse et element ubetydelig.

Det første element som skal deklareres er derfor rodelementet. Navnet på elementet skal anføres og dernæst, anført i parentes, den eller de knuder i strukturen som er indlejret i rodelementet.

I det anførte eksempel er der kun ét indlejret element: elementet BOG. Der er mindst én forekomst, men forhåbentlig mange flere, af dette element. Det angives med en **hyppighedsoperator**: + som betyder: én eller flere forekomster af det element som står umiddelbart foran. Ved at bruge operatorer for hyppighed, kan man angive hvor mange gange et element skal forekomme. Står der ikke en operator efter et element, forekommer det én og kun én gang.

Vi kan læse notationen i fig.3 på følgende måde: elementet ONLINE_KATALOG er forældreknude (eng. *parent*) til BOG, som er barn af (eng. *child*) ONLINE_KATALOG. BOG forekommer én eller flere gange.

3.1.2 Strukturelementet

Elementet BOG er et strukturelement. Det fremgår af indholdet i parentesen som kun indeholder elementer. Det deklareres i princippet efter samme opskrift som ovenfor:

<!ELEMENT bog (forfatter, impressum, stof) >

Dette element har som vist tre indlejrede elementer, de elementer som er anført i parentesen. Der er i dette eksempel ikke nogen hyppighedsoperator anført for disse elementer, og det betyder at de forekommer én – og kun én gang. Rækkefølgen af elementer i parentesen foreskriver rækkefølgen af elementer i XML-dokumentet. Vi kan læse notationen på følgende måde: elementet BOG er forældreknude til børnene FORFATTER, IMPRESSUM og STOF. Disse tre elementer er børn af BOG og de er tillige søskende (eng. *siblings*).

3.1.3 Tekstelementet

Som eksempel på et tekstelement kan vi bruge elementet FORNAVN:

<! ELEMENT fornavn (#PCDATA) >

Deklarationen siger i dette eksempel at elementet FORNAVN indeholder #PCDATA. Det betyder at indholdet i elementet er ren tekst. **PCDATA** betyder *Parsed Character Data*, data som skal undersøges nærmere, parses, med henblik på at sikre at der ikke forekommer ulovlige tegn i teksten, det vil sige tegn som indgår i XMLs standardvokabular.

3.1.4 Attributter

Der gælder en særlig syntaks for angivelse af attributter i en DTD. De bygges over følgende skabelon:

<!ATTLIST *elementnavn attributnavn typeværdi forekomst* **>**

Et eller flere attributter anføres i en **attributliste**: ATTLIST. Som det første angives hvilket element attributtet er associeret med, dernæst navnet på attributtet, hvilken type der optræder som værdi for attributtet, og endelig

en karakteristik af forekomsten, om der for eksempel er tale om et obligatorisk eller optionelt attribut.

Elementet FORFATTER indeholder et attribut:

<!ELEMENT forfatter (fornavn, efternavn) >

<!ATTLIST forfatter

nation CDATA #REQUIRED

>

FORFATTER er forældreknude til FORNAVN og EFTERNAVN, som dermed også er søskende. FORFATTER er med andre ord et *strukturelement*. Men derudover kan vi se at der er et attribut knyttet til elementet. Det angives med en attributliste: ATTLIST – det kan jo tænkes at der er mere end et attribut – efterfulgt af navnet på det element som indeholder et attribut efterfulgt af navnet på selve attributtet. Af ovenstående fremgår altså at elementet FORFATTER har et attribut kaldet NATION.

Herefter skal der følge yderligere to angivelser: en angivelse af hvilken type oplysning der kan forekomme som værdi for attributtet, og en angivelse af hvornår attributtets værdi skal være udfyldt.

I det aktuelle eksempel: **nation CDATA #REQUIRED,** står der umiddelbart efter attributtets navn som angivelse af typeværdi: **CDATA**. Denne oplysning betyder *Character Data*. Det angiver at der er tale om data som – i modsætning til PCDATA – ikke skal parses af XML-processoren (hvad det vil sige, vender vi tilbage til i modul 4). Som sidste oplysning finder vi angivelsen: **#REQUIRED**. Det betyder at der *skal* være en værdi angivet for dette attribut, værdiangivelsen er med andre ord obligatorisk, og der kan ikke forekomme en forfatter i vores onlinekatalog uden at der samtidig er en oplysning om vedkommendes nationalitet.

Et eksempel mere:

<!ELEMENT ISBN (#PCDATA) >

<!ATTLIST ISBN

format CDATA (tryk | lydbog) “tryk”

volumen CDATA #REQUIRED

>

Elementet ISBN er et tekstelement. Det er angivet som: ISBN (#PCDATA), hvor PCDATA betegner data som parses, nemlig tekststrengen som angiver ISBN nummeret. Der må med andre ord ikke være tegn indeholdt i denne streng som er reserveret til brug i XML.

Elementet ISBN har desuden, som det fremgår af ATTLIST, to attributter, attributterne FORMAT og VOLUMEN. For begges vedkommende gælder at typeværdien er defineret til at være character data. Men der er en væsentlig forskel i angivelsen af værdiens forekomst. Som det umiddelbart fremgår, skal der anføres en værdi for attributet VOLUMEN – i XML-dokumentet vil det være en angivelse af enten et sidetal eller en lydbogs varighed. Denne værdi er obligatorisk, hvad der fremgår af notationen #REQUIRED.

Værdien for attributet FORMAT er derimod angivet som en oplistning af mulige værdier (eng. *enumerated type values*): **format CDATA (tryk | lydbog) “tryk”**

Værdien kan være enten *tryk* eller *lydbog* svarende til angivelsen i parentesen: (tryk | lydbog), altså en bog i enten papirformat eller en lydbog. Endvidere er der angivet en default-værdi, det vil sige en værdi som automatisk vil gælde, hvis opmærkeren har glemt at anføre en værdi. Default er tryk, hvad der nok er det mest almindelige, trods alt.

Vi kan konstatere at vi nu er i stand til at bygge en DTD med baggrund i en formaliseret informationsmodel.

Den samlede DTD ser ud som nedenfor vist i (ex3):

Figur

```
          <!ELEMENT online_katalog ((bog+))>
          <!ELEMENT bog ((forfatter, impressum,
stof))>
          <!ELEMENT forfatter ((fornavn,
efternavn))>
          <!ATTLIST forfatter
          nation CDATA #REQUIRED
          >
          <!ELEMENT fornavn (#PCDATA)>
          <!ELEMENT efternavn (#PCDATA)>
          <!ELEMENT impressum ((titel, forlag,
udgivet, ISBN, pris))>
          <!ELEMENT titel (#PCDATA)>
          <!ELEMENT forlag (#PCDATA)>
          <!ATTLIST forlag
          sted CDATA #REQUIRED
          >
          <!ELEMENT udgivet (#PCDATA)>
          <!ELEMENT ISBN (#PCDATA)>
          <!ATTLIST ISBN
          format (tryk | lydbog) "tryk"
          volumen CDATA #REQUIRED
          >
          <!ELEMENT pris (#PCDATA)>
          <!ELEMENT stof ((handling, genre,
original_titel))>
          <!ELEMENT handling (#PCDATA)>
          <!ELEMENT genre (#PCDATA)>
          <!ELEMENT original_titel (#PCDATA)>
```

4. Validering af et XML-dokument med en DTD

Som det sidste skal vi se på hvordan man kan skabe en forbindelse mellem en DTD og et tilhørende XML-dokument således at det bliver muligt at validere opmærkningen i dokumentet.

Et XML-dokument er gyldigt hvis det matcher beskrivelsen af elementstruktur og indhold i en tilhørende DTD. Det vil med andre ord sige, at elementernes navne matcher, at rækkefølgen de forekommer i er den samme som beskrevet i DTD’en, og at de forekommer med samme hyppighed som angivet i DTD’en.

En DTD kan være **intern** i forhold til et XML-dokument eller den kan være **ekstern**.

1. En intern DTD er placeret øverst i XML-dokumentet, DTD og opmærkning er i et og samme dokument eller i en og samme fil.
2. En ekstern DTD opbevares i et dokument for sig, i en særskilt fil med efternavnet *.dtd, for eksempel online.dtd. Den DTD som er vist ovenfor i fig. 3 er et eksempel på en ekstern DTD.

Om man nu vælger at bruge en intern eller en ekstern DTD, afhænger af mange ting. Vælger man en intern DTD, følges DTD og opmærkning altid ad. Det kan undertiden være praktisk, specielt hvis der er tale om mindre XML-dokumenter som er afsluttede helheder og som ikke skal udbygges yderligere. Vælger man en ekstern DTD, opnår man mange fordele. Først og fremmest kan man distribuere sin DTD og dermed muliggøre at flere personer kan sidde samtidigt forskellige steder og gennemføre en ensartet og konsistent opmærkning af dokumenter, som afslutningsvis kan samles i ét større dokument.

Forbindelsen mellem et *XML-dokument* og *en ekstern DTD* etableres i den del af dokumentet som også kaldes for prologen. Det er her i et XML-dokument at man finder alle deklarationerne. Det kommer til at se således ud:

1: <?xml version="1.0" encoding="UTF-8"?>

2: <!DOCTYPE online_katalog

SYSTEM "d:\home\data\Altova Projects\online.dtd">

I første linie har vi den traditionelle XML-erklæring. I linie 2 følger et navn samt en angivelse af at der er en DTD knyttet til dette XML-dokument og at denne ligger på systemniveau: SYSTEM, og at den findes på det sted som er angivet i den efterfølgende sti. Læg mærke til at denne deklaration følger den traditionelle PI-syntaks: <!......> og at den indeholder to reserverede ord: DOCTYPE og SYSTEM.

Læg endelig som det sidste mærke til at vores DTD har et internt navn: **online_katalog**, det navn som står lige efter DOCTYPE, og at **dette navn er og skal være identisk med navnet på rodelementet i XML-dokumentet**.

Efter denne prolog følger resten af XML-dokumentet. Det kan nu udvides med flere forekomster af elementet BOG, og hver gang vi åbner dokumentet med en browser eller en anden form for XML-processor, bliver opmærkningen valideret. Det kan gøres for hver enkelt opmærkning decentralt og for hele kataloget centralt. Derfor er det en god idé at bruge en ekstern DTD!

Det nuværende dokument, klar til udvidelser og validering, ser således ud (ex.4):

Figur

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE online_katalog
SYSTEM "d:\home\data\Altova
Projects\nynetbog_online.dtd">

<online_katalog>

  <bog>
```

```
    <forfatter nation="dk">
      <fornavn>Jørgen</fornavn>
      <efternavn>Leth</efternavn>
    </forfatter>
    <impressum>
      <titel>Det uperfekte menneske</titel>
      <forlag sted="København">Gyldendal</forlag>
      <udgivet>2007</udgivet>
      <ISBN format="lydbog" volumen="11T:36M">978-87-02-06068-3</ISBN>
      <pris>kr. 298</pris>
    </impressum>
    <stof>
      <handling>Cool, elegante, morsomme og selvudslettende
                      historier fra et uperfekt liv med litteratur,
                      film, sport, kvinder og rejser. Digteren,
                      filminstruktøren og Tour de France-kommentatoren
                      m.m. har skrevet en erindringsbog af de sjældne,
                      som samtidig tegner et
```

```
portræt af dansk
                      kulturliv gennem de sidste
40 år. Jørgen Leths
                      karakteristiske stemme fører
lytteren gennem en
                      af dette årtis mest omtalte
bøger.</handling>

      <genre>biografi</genre>

      <original_titel> Det uperfekte Menneske
</original_titel>

    </stof>

  </bog>

</online_katalog>
```

Test dig selv

Vores XML-dokument,online_katalog.xml, og vores formaliserede informationsmodel repræsenterer en og samme trætruktur.

Hvad vil det sige at der er tale om en træstruktur?

I hvilken forstand kan man sige at der er tale om en repræsentation af en og samme træstruktur?

Tegn den træstruktur som er implementeret i de tre forskellige repræsentationer

Visning af XML med CSS
En introduktion til styling af XML med CSS

Styling XML med CSS

Et XML-dokument indeholder data opmærket i XML, det vil sige i et sprog som overholder den syntaks som gælder for velformet XML. En fornuftig navngivning af tags gør det muligt for et menneske at læse og i vid udstrækning at forstå den information som er indeholdt i et XML-dokument. Normalt vælger man derfor sine tags i opmærkningen så de beskriver indholdet i de data der opmærkes.

En tekststreng som: ’Skrjabin’ siger ikke de fleste noget. Men hvis man i et XML-dokument støder på et element som: <navn> Skrjabin </navn>, så er man formentlig straks klar over at der er tale om et navn. Og hvis man i et dokument støder på:

Figur `<komponist> <navn> Skrjabin </navn> </komponist>`

er det nu helt klart at der er tale om navnet på en komponist. På denne måde kan man med sin opmærkning gøre rå data, en tekststreng, til information, til navnet på en komponist.

Vi vil gerne præsentere denne information på skærmen i et brugervenligt format, uden tags og let at læse. Det kan vi gøre ved at udvikle og skrive et program, et såkaldt *stylesheet*.

Et style-sheet er en fil som indeholder en række instrukser om hvordan elementerne i et XML-dokument skal formatteres og præsenteres. Vi skal se nærmere på et særligt sprog, **Cascading Style Sheets**, forkortet **CSS**, som kan bruges til at beskrive hvordan man ønsker indholdet i et XML-dokument præsenteret af en browser. Men først lidt XML kode som vi kan bruge som eksempel. CD-XML1:

Figur

```
<cd> <komponist>Wolfgang Amadeus Mozart</komponist> <værk> Violinkoncert Nr.3 G-dur </værk> <opførelse> <orkester>Berliner Philharmoniker</orkester> <dirigent>Herbert von Karajan</dirigent> <solist>Anne-Sophie Mutter</solist> </opførelse> <indspilning>DGG 1978</indspilning> </cd>
```

Denne kode kunne være en del af et cd-onlinekatalog. Vores ambitionen er at give en potentiel kunde en anstændig præsentation af indholdet, som for eksempel:

Wolfgang Amadeus Mozart

Violinkoncert Nr.3 G-dur

Berliner Philharmoniker

Herbert von Karajan

Anne-Sophie Mutter

DGG 1978

Styling med CSS

Et stylesheet i CSS består af en eller flere **regler**. En regel i CSS skal være velformet, det vil sige at den skal overholde den særlige syntaks som gælder for CSS.Hver regel definerer hvordan et eller flere elementer i et tilknyttet XML-dokument skal præsenteres. En velformet regel bygges som vist i R1:

Figur

```
komponist { display:block; font-weight: bold; font-size:16pt; }
```

Som det første i en regel anføres en **selector**. En selector angiver navnet på det element i XML-dokumentet som skal formatteres, i det aktuelle tilfælde altså elementet KOMONIST. Indholdet i reglen, instrukserne om hvad der skal ske med elementet, skal for alle reglers vedkommende anføres mellem en start-klammer: { , og en slut-klammer: }. Dette indhold kaldes også for en **deklarationsblok** og den består af en eller flere **deklarationer** eller – mere informativt – af instrukser.

En instruktion i en regel følger et ganske bestemt mønster. Den består af et **attribut** og en **værdi**. Attribut og værdi adskilles fra hinanden med et kolon, og hele instruksen afsluttes med et semikolon.

Med attributtet angiver man *hvilken type* formattering man ønsker at gennemføre, og med værdien angives *hvordan* formatteringen skal gennemføres. Reglen R1 ovenfor indeholder således følgende instrukser:

1: display : block; indsæt linieskift før og efter elementets indhold

2: font-weight : bold; vis indholdet i bold

3: font-size : 16pt; sæt tegnstørrelsen til 16

Med regel R1 får vi med andre ord vist navnet på komponisten i bold, med et linieskift før og efter teksten. Læg mærke til at reglen viser hele indholdet i elementet KOMPONIST, vi behøver ikke at angive at der er tale om et navn.

Man kan i en selector også angive indtil flere elementer som skal formatteres ens, som vist med regel R2:

Figur `orkester,dirigent,solist { display:block; font-size:12pt; }`

Denne regel vil formattere indholdet i elementerne ORKESTER, DIRIGENT, SOLIST ud fra instrukserne angivet i deklarationsblokken: der skal et linieskift ind før og efter hvert element, og indholdet i alle elementer

skal skrives med tegnstørrelsen 12, ganske som vist i eksemplet ovenfor. Læg mærke til at vi ikke behøver at angive at elementerne er placeret i elementet OPFØRELSE.

Der er mange muligheder for at variere præsentationen. En blandt flere er at angive en bestemt farve på skriften. En anden udgave af R1:

Figur
```
komponist { display:block; font-weight: bold; font-size:16pt; color:red; }
```

vil resultere i at navnet på komponisten skrives med *rød* skrift, og en udvidelse af R2:

Figur
```
orkester,dirigent,solist { display:block; font-size:12pt; color:blue; }
```

vil medføre at teksten i de anførte elementer fremtræder i blå skrift.

CSS er et effektiv sprog som er ret enkelt at arbejde med. Det har dog også nogle ulemper, som man skal være opmærksom på. Vil du vide mere om CSS kan du kigger her: www.w3.org/TR/REC-CSS1 .

Linking XML og CSS

For at få XML og CSS til at samarbejde, skal man linke sit stylesheet til det XML-dokument som skal formatteres. Formålet med dette er at få den browser, man anvender, til at fortolke reglerne i det tilknyttede stylesheet og bruge dem på XML-dokumentet. Vi anvender CD-XML1 som dokument:

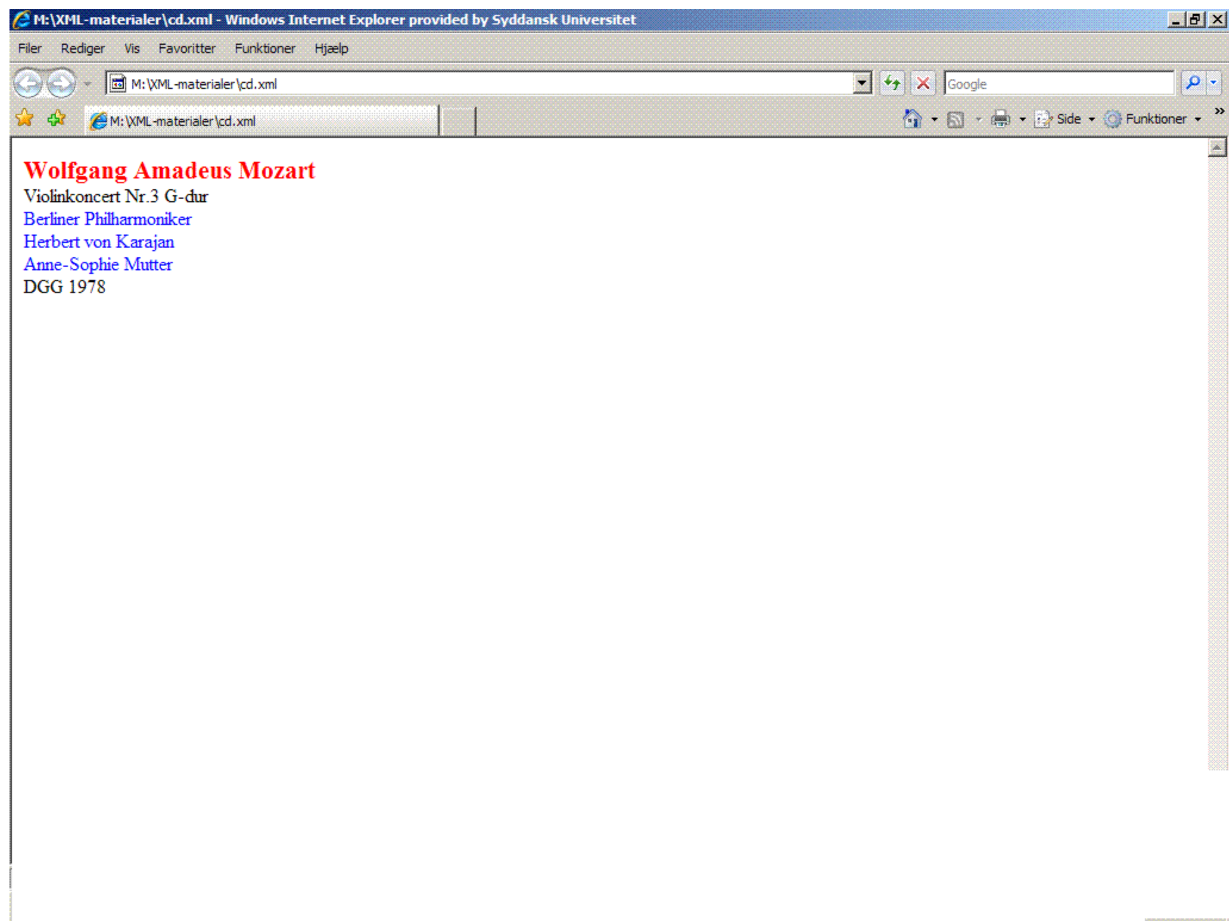
Figur
```
<?xml version="1.0" encoding="UTF-8"?> <?xml-stylesheet type="text/css" href="cd_onlinekatalog.css"?> <cd> <komponist>Wolfgang Amadeus Mozart</komponist>
```

```
<værk> Violinkoncert Nr.3 G-dur </værk>
<opførelse> <orkester>Berliner
Philharmoniker</orkester> <dirigent>Herbert von
Karajan</dirigent> <solist>Anne-Sophie
Mutter</solist> </opførelse> <indspilning>DGG
1978</indspilning> </cd>
```

Som det fremgår, er der nu tilføjet en prolog, hvor vi (1) deklarerer den version af XML som anvendes og det tegnsæt som skal bruges, og (2) skaber en forbindelse til det stylesheet som skal bruges:

<?xml-stylesheet type="text/css" href="cd_katalog.css"?>

Her angives at typen er en tekstfil indeholdende CSS, og vi angiver en *adresse* på filen og dens navn. I dette tilfælde ligger XML-dokumentet og vores CSS stylesheet i samme mappe. Med tilføjelsen af denne prolog vil et klik på XML-dokumentet give følgende resultat:



Vil du vide mere om de mange muligheder for at udnytte forskellige former for styling med CSS og hvordan man kan kombinere CSS og HTML, er der flere muligheder: www.w3.org/MarkUp/Guide/Style eller www.w3.org/Style/CSS/learning .

Fra XML til HTML med XSLT – en introduktion

Introduktion

I modsætning til HTML-dokumenter, der automatisk bliver vist med typografi og layout, når de åbnes i en browser som Internet Explorer eller Firefox, indeholder XML-dokumenter i sig selv normalt ingen information om, hvordan de skal visuelt skal fremstå, og vil derfor blot blive præsenteret i en form for ”rå kode”, når man åbner dem i en browser. Dette er helt tilsigtet: ideen med XML er blandt andet, at et XML-dokument skal kunne vises forskelligt, alt efter hvem målgruppen er, hvad konteksten er, eller hvilket medium dokumentet skal vises i. Og nogle gange skal et XML-dokument slet ikke præsenteres. Det gælder f.eks. i de tilfælde, hvor XML bliver brugt som format for udveksling af data mellem it-systemer, og hvor det overhovedet ikke er nødvendigt, at XML-dokumentet læses af et menneske. I andre tilfælde er en læsevenlig præsentation af XML-dokumenter helt nødvendig, f.eks. hvis de skal publiceres på Nettet og læses af en bruger i en browser.

Præsentation med CSS

Man kan give et XML-dokument en visuel form på forskellige måder. En ofte anvendt metode er at koble et CSS-stylesheet til XML-dokumentet. Et CSS-style sheet er et simpelt tekstdokument, der indeholder en række regler, som foreskriver, hvordan indholdet i XML-dokumentet typografisk og layoutmæssigt skal fremstå. I og med at et XML-dokument kan beskrives som en træstruktur, taler man nogle gange om, at et CSS-stylesheet ”pynter” XML-træet. CSS-style sheets er forholdsvis enkle og nemme at udarbejde, men har også visse begrænsninger. Eksempelvis er det vanskeligt med et CSS-style sheet at vise et XML-dokuments enkelte indholdselementer i en anden rækkefølge end den, de optræder i, i selve XML-dokumentet.

XSLT-stylesheets

Langt mere fleksible er XSLT-style sheets. Med et XSLT-style sheet kan man udtrække og vise XML-indhold på et utal af måder, til forskellige formål, og til diverse medier som f.eks. computerskærm eller mobiltelefon.

XSLT, en forkortelse for Extensible Style Language Transformations, er en slags programmeringssprog til behandling af XML-dokumenter.

Programmer udarbejdet ved hjælp af XSLT kaldes for XSLT-scripts, XSLT-transforms eller XSLT-style sheets.

Et XSLT-style sheet er, lige som et CSS-style sheet, et tekstdokument med et sæt af regler. Disse regler er imidlertid ikke regler, som direkte beskriver, hvordan en given datastruktur i et XML-dokument skal præsenteres, men et sæt instrukser, der specificerer, hvordan denne datastruktur skal transformeres. Med et XSLT-style sheet kan man transformere et dokument i et XML-format til et andet; man kan lave det om til HTML, eller man kan konvertere det til en almindelig tekst.

Transformation af XML med XSLT
Et XML-dokument, der skal vises på Nettet, skal typisk transformeres til HTML eller XHTML, således at det kan vises i en browser. Rent konkret foregår en XML-til-HTML transformation ved, at en XML-processor, et stykke software, læser XSLT-style sheetet og gennemfører transformationen af XML-dokumentet, også kaldet kildedokumentet, som er foreskrevet i XSLT-style sheetet. Transformationen kan ske på en webserver (”server side”), inden det sendes af sted til eller i slutbrugerens egen browser (”client side”). I begge tilfælde bliver der som output genereret (X)HTML-kode, som kan vises i en browser. Selve koblingen af XML-dokumentet og det tilhørende XSLT-stylesheet kan ske ad flere veje. Meget ofte vil der simpelthen være et link fra kildedokumentet til det style sheet, som XML-processoren skal benytte i forbindelse med transformationen.

(Se en visualisering af, hvordan en XSLT-transformation foregår).

Et XSLT-style sheet kan som sagt opfattes som en samling af transformationsregler. En transformationsregel omfatter normalt to instrukser: en instruks, der angiver, hvad der skal transformeres i kildedokumentet og en anden instruks, der specificerer, hvad resultatet af transformationen skal være. En transformationsregel til et bogkatalog i XML, der skal præsenteres på Nettet, kunne eksempelvis på (nogenlunde) mundret dansk lyde sådan her:

Note: Find alle krimiforfattere og præsenter dem i en nummereret liste.

I en XML-til-HTML transformation skulle denne instruks måske formuleres som:

> **Note:** Find indholdet af elementet **navn** i alle **forfatter**-elementer, som findes i **krimi**-elementet i kildedokumentet og sæt det ind i et **li**-element i en **ol**.

En sådan instruktion ville med andre ord konvertere en XML-datastruktur som: `<krimi> <forfatter> <navn>Jens Hansen</navn> </forfatter> <forfatter> <navn>Hans Jensen</navn> </forfatter> </krimi>` til følgende HTML-output: `<ol> <li>Jens Hansen</li> <li>Hans Jensen</li> </ol>`

Øvelse
Hvordan transformationsregler rent faktisk kodes i XSLT, og hvordan et XSLT-style sheet egentlig er opbygget, illustreres nedenfor gennem et konkret eksempel. Eksemplet indeholder et kildedokument i XML (en pressemeddelelse), et XSLT-style sheet og den HTML-kode, som genereres, når style sheetet appliceres på kildedokumentet i en XML-processor. Kig på kildedokument, style sheet og output og besvar de tilhørende spørgsmål: `<?xml version="1.0"?> <?xml-stylesheet type="text/xsl" href="pressemeddelelse1.xsl"?> <pressemeddelelse> <dato>12.12.2008</dato> <overskrift>Ny dokumentstandard i Opdikt A/S </overskrift> <resume>Opdikt A/S har i dag offentliggjort planer om at indføre XML som fælles dokumentstandard i hele organisationen</resume> <indhold>På direktionens møde i dag blev det besluttet at indføre XML som fælles format i virksomhedens Web-baserede kommunikation. XML vil blive anvendt på virksomhedens intranet, i den Web-baserede markedskommunikation og, sidst men ikke mindst, som dataudvekslingsformat i`

```
forbindelse med e-handelstransaktioner.
Implementeringen af planen varetages af
virksomhedens IT-afdeling og forventes at løbe et
halvt år. </indhold> <kontaktperson>
<fornavn>Hans</fornavn>
<efternavn>Jensen</efternavn> <afdeling>It-
afdelingen</afdeling>
<email>hans@it.opdikt.dk</email> </kontaktperson>
</pressemeddelelse> <xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0"> <xsl:template match="/"> <html>
<head> <title>Pressemeddelelse</title> </head>
<body> <p style="font-
size:24">Pressemeddelelse</p> <p align="right">
<xsl:value-of select="pressemeddelelse/dato" />
</p> <p style="font-weight:bold; font-size:18">
<xsl:value-of select="pressemeddelelse/overskrift"
/></p> <p><i><xsl:value-of
select="pressemeddelelse/resume" /></i></p> <p>
<xsl:value-of select="pressemeddelelse/indhold" />
</p> <p><b>Kontakt:</b></p> <p><xsl:value-of
select="pressemeddelelse/kontaktperson/fornavn" />
<xsl:text> </xsl:text> <xsl:value-of
select="pressemeddelelse/kontaktperson/efternavn"
/> <xsl:text> </xsl:text> (email: <xsl:value-of
select="pressemeddelelse/kontaktperson/email" />)
</p> </body> </html> </xsl:template>
</xsl:stylesheet> <html> <head> <META http-
equiv="Content-Type" content="text/html;
charset=utf-16"> <title>Pressemeddelelse</title>
</head> <body> <p style="font-
size:24">Pressemeddelelse</p> <p
align="right">12.12.2008</p> <p style="font-
weight:bold; font-size:18">Ny dokumentstandard i
Opdikt A/S </p> <p><i>Opdikt A/S har i dag
offentliggjort planer om at indføre XML som fælles
```

```
dokumentstandard i hele organisationen</i></p> <p>
På direktionens møde i dag blev det besluttet at
indføre XML som fælles format i virksomhedens Web-
baserede kommunikation. XML vil blive anvendt på
virksomhedens intranet, i den Web-baserede
markedskommunikation og, sidst men ikke mindst,
som dataudvekslingsformat i forbindelse med e-
handelstransaktioner. Implementeringen af planen
varetages af virksomhedens IT-afdeling og
forventes at løbe et halvt år. </p> <p><b>Kontakt:
</b></p> <p>Hans Jensen (email: hans@it.opdikt.dk)
</p> </body> </html>
```

Spørgsmål

1. Hvordan kobles XML-dokument og XSLT-style sheet sammen?
2. Kildedokumentet er i XML. Hvad med XSLT-style sheetet?
3. Nogle elementer i style sheetet starter med ”forstavelsen” **xsl**, men andre ikke gør. Hvorfor mon?
4. Hvilken instruks bruges i style sheetet til at udtrække data fra kildedokumentet?
5. Hvordan lokaliseres det indhold i kildedokumentet, der skal udtrækkes?
6. Man taler nogle gange om, at XML-indhold ”pakkes ind” i HTML. Forklar denne metafor.
7. Bliver alt indhold vist i HTML-outputtet? Hvorfor/hvorfor ikke?

Hvad består et XML-til-HTML XSLT-stylesheet af?

Introduktion

Et **XSLT-stylesheet** kan beskrives som en samling af regler, der beskriver, hvordan et (sæt af) XML-dokument(er) skal konverteres. Mere præcist foreskriver et XSLT-stylesheet, hvordan en given XML-struktur skal **transformeres** til en anden informationsstruktur. Denne struktur kan være en anden XML-struktur, det kan være HTML, eller det kan være almindelig tekst. En transformation ved hjælp af XSLT kan også beskrives som et udtræk af udvalgte XML-data og indsættelse af disse data i et nyt dokument.

XSLT-stylesheets bruges meget ofte til at konvertere XML-dokumenter til HTML, således at disse kan vises som websider. Et XSLT-stylesheet til transformation af XML til HTML kunne f.eks. være nedenstående, som specificerer, hvordan data i et "XML-kildedokument" skal udtrækkes og vises i HTML. (Kildedokumentet kan ses her.)

```
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:template match="/">
    <html>
      <head>
        <title>Pressemeddelelse</title>
       </head>
      <body>
       <p style="font-
size:24">Pressemeddelelse</p>
       <p align="right"><xsl:value-of
select="pressemeddelelse/dato" /></p>
       <p style="font-weight:bold; font-size:18">
         <xsl:value-of
select="pressemeddelelse/overskrift" /></p>
       <p><i><xsl:value-of
```

```
select="pressemeddelelse/resume" /></i></p>
       <p><xsl:value-of
select="pressemeddelelse/indhold" /></p>
       <p><b>Kontakt:</b></p>
       <p><xsl:value-of
select="pressemeddelelse/kontaktperson/fornavn" />
         <xsl:text> </xsl:text>
       <xsl:value-of
select="pressemeddelelse/kontaktperson/efternavn"
/>
         <xsl:text> </xsl:text>
      (email: <xsl:value-of
select="pressemeddelelse/kontaktperson/email" />)
</p>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Rodelement
Det første man skal bemærke er, at et XSLT-stylesheet i sig selv er et XML-dokument, selv om dette eksempel rent faktisk ikke indeholder en **XML-deklaration**. Alle XML-dokumenter har som bekendt et **rodelement**, det element, der indeholder alle de andre elementer, og det gælder således også for XSLT-stylesheets. I et XSLT-stylesheet er rodelementet normalt `<xsl:stylesheet>...</xsl:stylesheet>`. Rodelementet indeholder, udover en versionsangivelse, også en såkaldt **name space deklaration**, nemlig `xmlns:xsl="http://www.w3.org/1999/XSL/Transform"`. Denne kode har blandt andet til formål at identificere alle de elementer i stylesheetet, som tilhører "XSLT name space", dvs. elementer, som har en særlig betydning eller funktion i XSLT. Rent konkret drejer det sig om alle de elementer, der har **præfixet** "xsl". Disse elementer udgør de egentlige instruktioner i programmet og omfatter foruden selve rodelementet eksempelvis `<xsl:template>` og `<xsl:value-of>`.

xsl:template
<xsl:template>-elementet omfatter et sæt af sammenhørende transformationsregler kaldet en **template rule**. I ovennævnte style sheet er der kun en template rule, men almindeligvis vil et XSLT-style sheet indeholde flere. <xsl:template>-elementet har en attribut, nemlig "match", der angiver den del af kildedokumentet, som skal transformeres. I dette tilfælde er værdien af attributten "/", hvilket lidt firkantet udtrykt betyder, at "hele kildedokumentet" skal behandles.

xsl:value-of
Et andet centralt xsl-element er <xsl:value-of>-elementet. Dette element bruges i XSLT til at angive hvilke data, der skal udtrækkes fra kildedokumentet. Selve identifikationen af data sker gennem attributten "select", der udpeger de relevante data og deres placering i kildedokumentet Et eksempel er f.eks. select="pressemeddelelse/kontaktperson/efternavn". Med denne værdi anføres det altså, at indholdet af elementet "efternavn", som er underelement til elementet "kontaktperson", der igen er underelement til "pressemeddelelse", skal udtrækkes.

Literal result
Mens elementer med præfixet "xsl" altså er de elementer i style sheetet, som angiver instruktioner i forbindelse med selve transformationen, angiver alle de andre **the literal result**, dvs. det output, som skal genereres. I eksemplet her er det med andre ord al HTML-koden så som (<html>, <body>, <p> og så videre. Selve koblingen mellem xsl-elementer og outputelementer sker ved, at de indlejres i hinanden. Et eksempel:

```
         <p><i><xsl:value-of
select="pressemeddelelse/resume" /></i></p>
```

Her er xsl-instruktionen <xsl:value-of select="pressemeddelelse/resume"> indlejret i HTML-koden

`<p><i>...</i></p>`. Det betyder konkret, at indholdet af elementet "resume", der ligger i elementet "pressemeddelelse" udtrækkes fra kildedokumentet og præsenteres i et kursiveret afsnit. Med en metafor kan man sige, at XML-data udtrækkes og "pakkes ind" i HTML.